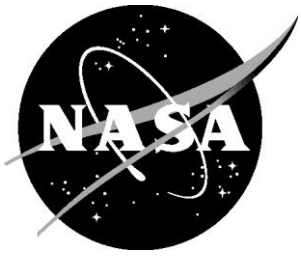
NASA/TM–2015-218783



Overview of Threats and Failure Models for Safety-Relevant Computer-Based Systems

Wilfredo Torres-Pomales
Langley Research Center, Hampton, Virginia

September 2015

NASA STI Program . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA's STI. The NASA STI program provides access to the NTRS Registered and its public interface, the NASA Technical Reports Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counter-part of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing information desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

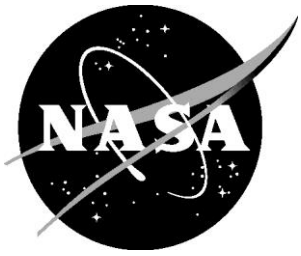
- Access the NASA STI program home page at http://www.sti.nasa.gov

- E-mail your question to help@sti.nasa.gov

- Phone the NASA STI Information Desk at 757-864-9658

- Write to:
  NASA STI Information Desk
  Mail Stop 148
  NASA Langley Research Center
  Hampton, VA 23681-2199

NASA/TM–2015-218783



Overview of Threats and Failure Models for Safety-Relevant Computer-Based Systems

Wilfredo Torres-Pomales
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

September 2015

Acknowledgments

I would like to express my gratitude to the reviewers and to those who have helped me gain a better appreciation of the complexities in the design and evaluation of dependable systems.

Available from:

NASA STI Program / Mail Stop 148
NASA Langley Research Center
Hampton, VA 23681-2199
Fax: 757-864-6500

Abstract

This document presents a high-level overview of the threats to safety-relevant computer-based systems, including (1) a description of the introduction and activation of physical and logical faults, (2) the propagation of their effects, and (3) function-level and component-level error and failure mode models. These models can be used in the definition of fault hypotheses (i.e., assumptions) for threat-risk mitigation strategies. This document is a contribution to a guide currently under development that is intended to provide a general technical foundation for designers and evaluators of safety-relevant systems.

Table of Contents

Abbreviations v

1. Introduction 1

2. System Safety Threats 2
   2.1. Threats-and-Effects Model 2
   2.2. Fault Introduction 5
      2.2.1. Non-Operational Threats 6
      2.2.2. Operational Threats 7
      2.2.3. Physical Faults 8
      2.2.4. Logical Faults 8
   2.3. Logical Fault Activation and Effects Propagation 8
   2.4. Functional Failure Model 9
      2.4.1. Source-Based versus Receiver-Based Failure Model 10
      2.4.2. Functional Error Model 10
      2.4.3. Functional Failure Mode Model 11
      2.4.4. Transition Model for Functional Failure Modes 12
   2.5. Component Failure Model 13
      2.5.1. Component Error Model for Single-User Service 13
      2.5.2. Component Failure Mode Model for Single-User Service 14
      2.5.3. Component Error Model for Multiple-User Service 15
      2.5.4. Component Failure Modes for Multiple-User Service 17
      2.5.5. Failure Models with Explicit Value and Time Dimensions 17

3. Final Remarks 19

References 20

Abbreviations

A_A	Acceptable-Item Asymmetry
A_n	User Asymmetry
C	Correct
ConOps	Concept of Operations
COTS	Commercial Off The Shelf
CS	Correct Symmetric
D	Detectable
DIMA	Distributed Integrated Modular Architecture
DMA	Direct Memory Access
DPS	Data Processing System
ESD	Electrostatic Discharge
FAA	Federal Aviation Administration
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
IEEE	Institute of Electrical and Electronics Engineers
IKIWISI	I'll-know-it-when-I-see-it
IMA	Integrated Modular Avionics
IoP	Index of Performance
IUE	Initiating Unintended Event
LoF	Loss of Function
LRU	Line Replaceable Unit
MF	Malfunction
MIL-STD	Military Standard
NAS	National Airspace
NASA	National Aeronautics and Space Administration
OS	Omissive Symmetric
OTH	Omissive-Transmissive Hybrid
RF	Radio Frequency
S_A	Acceptable-Item Symmetry
SDOA	Single-Data Omissive Asymmetric
S_n	User Symmetry
SOA	Strictly Omissive Asymmetric
SOI	System Of Interest
SS	Sub-System
TA	Transmissive Asymmetric
TRL	Technology Readiness Level
TS	Transmissive Symmetric
TTC	Time To Criticality
TTE	Time To Effect
TUE	Terminal Unintended Event
U	Undetectable
V&V	Validation and Verification

1. Introduction

An aircraft consists of a collection of systems performing a wide variety of functions with different safety criticality levels. The aviation industry is continuing a decades-old trend of adopting increasingly sophisticated computer-based technology to implement aircraft functionality. Modern aircraft are highly complex, functionally integrated, network-centric systems of systems [1]. The design and analysis of distributed-computation aircraft systems are inherently complex activities. Ensuring that such systems are safe and comply with existing airworthiness regulations is costly and time-consuming as the level of rigor in the development process, especially the validation and verification activities, is determined by considerations of system complexity and safety criticality. A significant degree of care and deep insight into the operational principles of these systems are necessary to ensure adequate coverage of all design implications relevant to system safety.

The validation and verification (V&V) and the certification of complex computer-based systems, including safety-critical systems, are recognized problems of national significance [2], [3], [4]. The challenges in assuring the design and safety of systems require considerable attention and financial investment [5]. As aircraft system complexity continues to increase, the V&V and certification costs and related programmatic risks can provide a basis against the development and implementation of new capabilities [6]. Such obstacles against innovation pose a threat to national competitiveness and can hinder the proposed operational improvements to the National Airspace System (NAS) that are intended to increase capacity and flexibility and reduce costs, but would also increase the complexity of airborne and ground aviation systems [7]. There are initiatives underway to produce methods, tools, and techniques that enable predictable, timely, and cost-effective complex systems development [8]. NASA aims to identify technical risks and to provide knowledge to safely manage the increasing complexity in the design and operation of vehicles in the air transportation system. For this goal, multidisciplinary tools and techniques are being developed to assess and ensure safety in complex aviation systems and enable needed improvements to the NAS.

This document is a contribution to a design and evaluation guide currently under development. The guide is intended to (1) provide insight into the system safety domain, (2) present a general technical foundation for designers and evaluators of safety-relevant systems, and (3) serve as a reference for designers to formulate well-reasoned safety-related claims and arguments and identify evidence that can substantiate the claims. That evidence forms a basis for demonstrating compliance with certification regulations. The generation of such evidence is a major objective of a system development process. This work is part of an ongoing effort to enable sound assurance of safety-related properties of computer-based aircraft systems by developing an effective capability to model and reason about the safety implications of system requirements and design.

This document provides a high-level overview of threats to safety-relevant computer-based systems, covering the introduction and activation of physical and logical faults and the propagation of their effects. Function-level and component-level error and failure mode models are presented. These models can be used in the definition of fault hypotheses (i.e., assumptions) for threat-risk mitigation strategies. That will be the topic of the next contribution to the design and evaluation guide.

2. System Safety Threats

System-safety threats are processes or phenomena that can cause operational hazards. The type of hazard considered in this paper is the failure of a given computational system, henceforth referred to as the system of interest (SOI). The failure threats can occur during development, production, manufacturing, operation, or maintenance of the SOI, and they can be endogenous or exogenous to the system. This section examines the relation between threats and failures and introduces generic functional failure models applicable at every level in a design hierarchy. These models can be used in the analysis of functional and non-functional (i.e., quality) properties of a system.

2.1. Threats-and-Effects Model

A system is an entity with a purpose achieved through the interactions of its internal components with each other and with the external environment. A system has a hierarchical structure in which the components are themselves systems (i.e., sub-systems) in a sequence of recursive composition that continues until primitive atomic components are reached whose internal structure is irrelevant or unknown. A system can be conceptualized as consisting of three layers (also referred to as universes or domains) [9]: **information** (i.e., data sets and streams), **logical** (i.e., computation and communication algorithms), and **physical** (i.e., concrete physical substrate). A layer is also referred to as a **domain** or **universe**. A system has a hierarchical organizational structure at each of these layer. The mapping of components between layers can be one-to-many, many-to-one, and/or one-to-one.

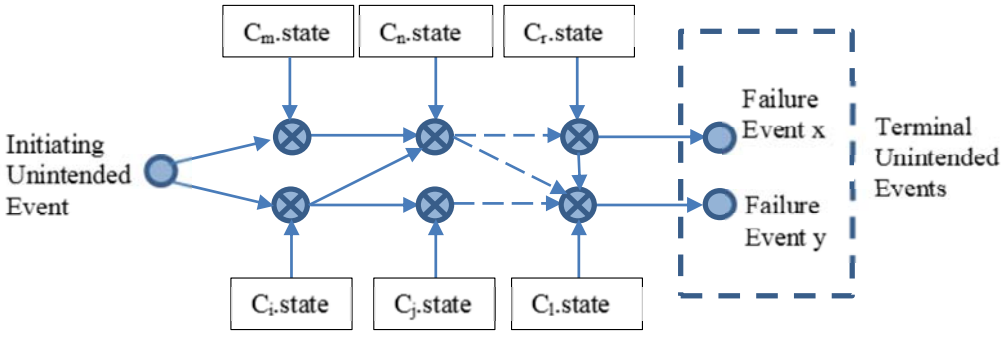


Figure 1: Notional Example of System Failure Model as an Event Chain

A generic failure model for computational systems is similar to a hazard scenario model [10] and is based on the concept of a causal chain of events. An event is a change in state at a specified location. Figure 1 illustrates a notional example of the system failure model, where $C_i.state$ represents the state of a component *i*. Under this failure model, **Initiating Unintended Events (IUE)** propagate effects through coupling paths until they reach system outputs as **Terminal Unintended Events (TUE)**. This conceptual system failure model is an abstraction of the propagation mechanism from IUEs to TUEs. The IUEs can originate within or outside the system boundary. The coupling paths can be either intended (i.e., in-band [11]) interaction paths defined by the system design, or unintended (i.e., out-of-band [11]) interaction paths that have been assumed not to exist in the design and description of the system (i.e., interaction paths that

violate coupling assumptions). The value and timing characteristics of propagated effects are dependent on the structure and state of the system. The effects may be delayed until the system state is favorable for propagation. An event is said to be **active** (or **live**) when it is propagating effects; otherwise, the event is said to be **dormant** (or **latent**) [12]. Under the event chain model, a system **failure** occurs when the delivered service is not as intended due the effects of IUEs. Note in Figure 1 that IUE effects can propagate simultaneously through multiple paths. In general, event chains do not necessarily propagate effects forward in linear patterns, but may actually have complex propagation patterns with divergent and convergent paths as well as forward and backward paths (e.g., consider state machines with feedback loops).

The general system structure and layers can be examined to gain further insight into this system failure model. Figure 2 illustrates an example of a failure event chain for a fragment of a hierarchical structure in the logical layer of a system. The clouds in Figure 2 represent effect propagation paths in the system. In this example, an IUE occurs at point A and its effects propagate to output point B when the state of component V is favorable for such propagation. A TUE at point B constitutes a failure of Component V. In Component W, the effects of the unintended event at point B may propagate if the state of component W enables it. Component W experiences a service failure if unintended events reach the output at point C.

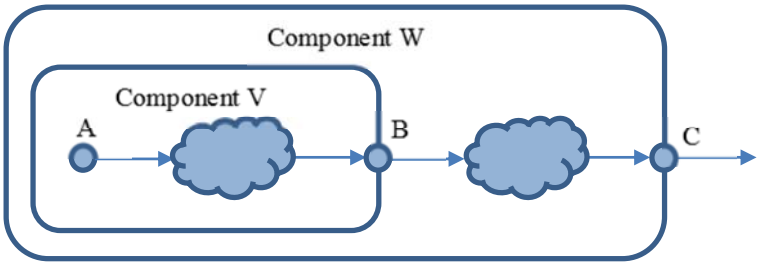
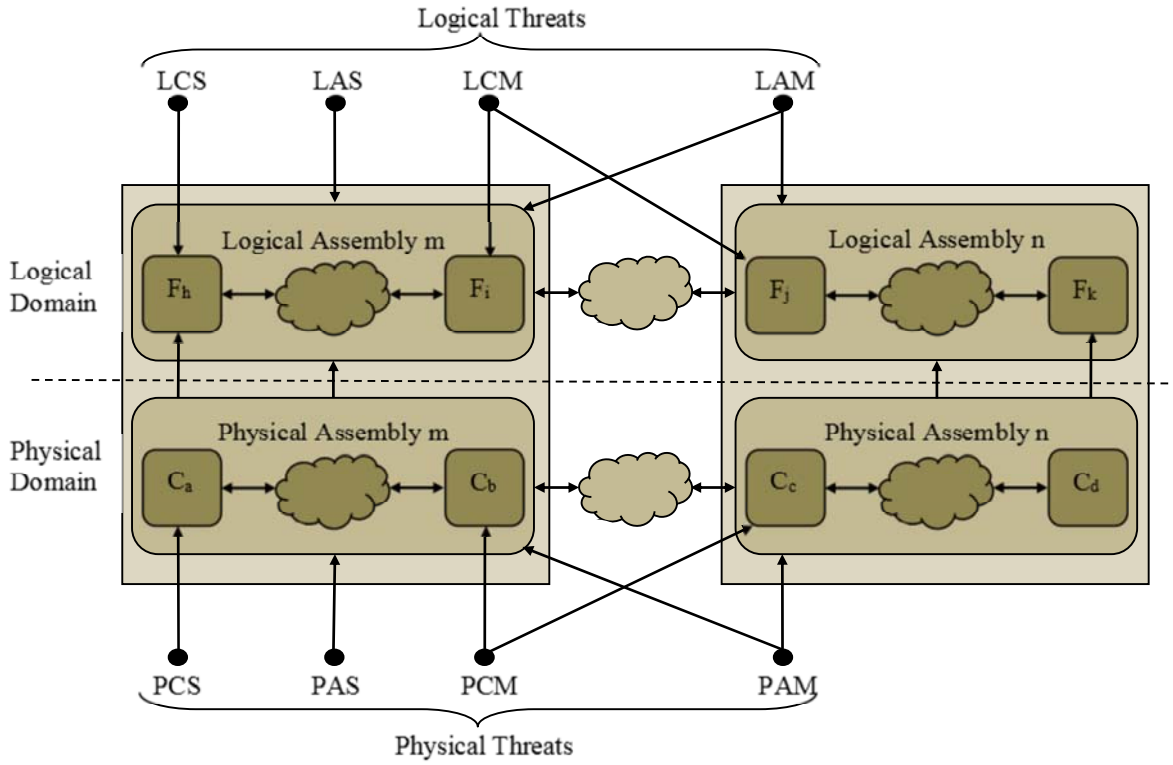


Figure 2: An Event Chain in a Fragment of a Logical Hierarchical Composition

In Figure 2, the event propagation mechanisms between points A, B, and C are **couplings** between these points. These define relations between events at points A and B, B and C, and A and C. Variability and uncertainties in these relations and in the states and events (i.e., state transitions) at initiating points influence the variability and uncertainties at terminal points. Hence, the system failure risk (i.e., severity and likelihood of failure events) can be mitigated by controlling the initiating events and the coupling mechanisms (both intended and unintended), including intermediate events between initiating and terminal event points. This insight will be explored further in future sections of the design and evaluation guide.

The **threats** of interest here are processes or phenomena that can cause system functional failures by introducing IUEs. These threats can be external or internal to the SOI. A functional failure occurs when the SOI does not deliver the intended service as reflected in the output data in the information layer. This can be caused by IUEs in the logical layer, or by IUEs in the physical layer that propagate effects through the logical layer and on to the information layer. An IUE is the result of a physical or logical **defect** caused by a threat. Such a physical or logical defect in a system is called a **fault**. Threats are fault sources (i.e., causes). We refer to threats that cause physical faults as **physical threats**, and those that cause logical faults as **logical threats**. Faults in the logical layer can cause **errors** in the information layer (i.e., data with unintended value and/or timing) that can propagate to system outputs where they manifest as failures. An error is essentially a fault (i.e., a defect) in the information layer. Avizienis et al. [13] and Parhami [14] [15] provide additional information on the fault-error-failure propagation model.

Figure 3 illustrates some possible fault introduction targets (or victims) of physical and logical threats. The oval-ended arrows identify direct targets of threats where faults are introduced. As in Figure 2, a cloud in Figure 3 represents a coupling mechanism between parts of a system. In this figure, a group of related components is called an **assembly**. The classification of fault introduction targets is based on three characterizing dimensions: **layer** (physical or logical), **part** (component or assembly), and **count** (single or multiple). A fault in a component afflicts that specific component, while a fault in an assembly afflicts the integration of all the contained components (i.e., the assembly as a whole is faulty rather than just parts of it). A threat can directly afflict individual or multiple parts, which can be either components or assemblies.



Threat Code: LPC		
L	Layer	Physical (P) or Logical (L)
P	Part	Component (C) or Assembly (A)
C	Count	Single (S) or Multiple (M)

Figure 3: Examples of Fault Introduction Targets from Physical and Logical Threats

As illustrated in Figure 3, in this model a threat can generate faults directly or indirectly (as a sequential or cascade propagation of effects) in one or more parts of the system. A fault directly caused by a threat is

a **primary fault**. A fault in one component or assembly can propagate effects through coupling mechanisms to other components or assemblies on the same layer. Physical faults can also propagate effects to the logical layer. Faults (i.e., defects) that occur as a result of propagation from one part to another are called **secondary faults**. **Common-cause** (i.e., **dependent** or **related**) **faults** in the physical or logical layers can be multiple primary faults caused by the same threat event, or a primary fault with its propagated secondary faults. Common-cause faults require special consideration in the design of threat mitigation strategies that are based on assumed bounds on the number of simultaneously active faults.

Additional information on fault and failure propagation modeling and analysis is available from a multitude of sources. SAE Aerospace Recommended Practice ARP-1834A [16] describes recommended practice for functional failure analysis of digital systems and equipment. Beland and BonJour [17] describe an approach for modeling and analysis of functional failure in airborne electronic hardware. ARP-4761 [18] describes approaches for the analysis of failure effects in aircraft systems. Leveson [19] provides an extensive and insightful presentation of system safety, including accident modeling and analysis. Most of the information available on common-cause failure is in the context of nuclear power plants. Documents from the International Atomic Energy Agency (IAEA) [20], U.S. Nuclear Regulatory Commission [21] [22] [23], Rutledge [24] [25], Stott et al. [26], and Kaufman et al. [27], among many others, provide insight into this type of fault and the mathematical models of their propagation mechanisms and effects.

2.2. Fault Introduction

In this section, we consider the (unintended) introduction of primary faults and the generation of secondary faults within the physical and logical layers or across from the physical layer to the logical layer. Primary faults can be introduced during development, production, manufacturing, operation, or maintenance of a system. We divide the threat and fault spaces along two dimensions: **life cycle phase** (non-operational or operational) and **layer** (physical or logical). The structure of the threat and fault sets (or spaces) is illustrated in Table 1 and Figure 4. **Non-operational faults** are caused by (non-operational) threats during development, production, manufacturing, refinement, or maintenance of a system. These can be logical or physical faults. **Operational faults** are caused by (operational) threats when the system is in operation. We assume here that the logic design of a safety-relevant system remains unchanged during operation. Hence, only physical primary faults can be introduced during operation, though the generation of secondary faults in the logical layer during operation is possible. Avizienis et al. [13], Suri et al. [28], Heimerdinger and Weinstock [29], and Hugue [30] propose additional fault classification criteria for more refined taxonomies.

Table 1: Structure of Threat and Fault Spaces

Threat and Fault Spaces		Layer	
		Logical	**Physical**
Life Cycle Phase	**Non-Operational**	Non-Operational Logical Threats and Faults	Non-Operational Physical Threats and Faults
	Operational	[Assumed Empty]	Operational Physical Threats and Faults

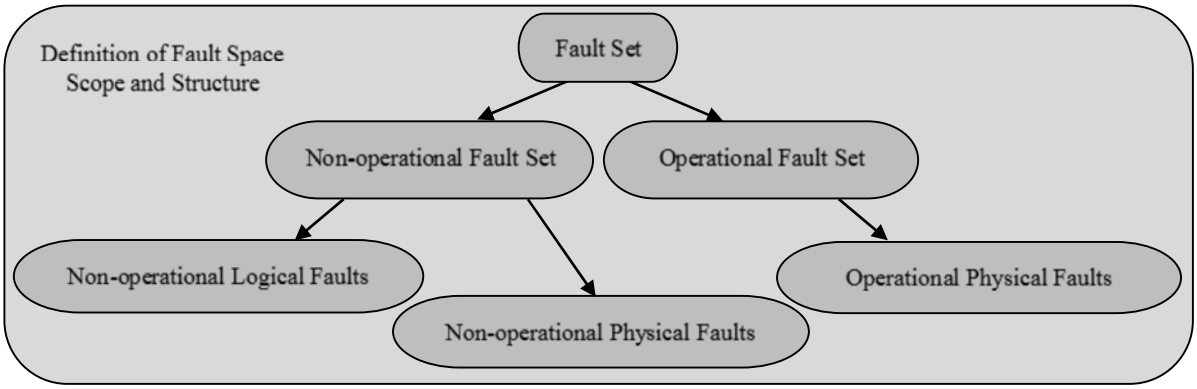


Figure 4: Structure of the System Fault Space

2.2.1. Non-Operational Threats

Non-operational threats are processes or phenomena that can introduce logical and physical faults into the system during non-operational phases of the life cycle. These phases include development, production, manufacturing, refinement, installation, and maintenance.

System development can be viewed as a process of making a hierarchy of decisions such that early high-level, broad-scope decisions provide a basis and context for later lower level and narrower-scope decisions. From this perspective, mistakes in early design decisions can propagate effects in the form of additional wrong decisions and faulty (i.e., defective) design and implementation to later stages of the development and implementation. These mistakes can affect the functional, physical, and allocated system architectures, as well as the hardware and software, and their integration. Design and implementation refinements after initial system deployment can introduce faults in a similar way as the original development. Faults introduced during development and refinement can be caused by, for example, incorrect or incomplete requirements, incorrect interpretation of requirements, inconsistent assumptions in the design of components, and random design and implementation mistakes. Note that system requirements result from

identified customer needs as well as development decisions, and defects in requirements can lead to defects in design (i.e., secondary development faults).

Manufacturing and production defects, particularly in hardware, can occur in every aspect of the product. This includes things such as packaging and soldering defects, selection of wrong materials, damage due to electrostatic discharge (ESD) or electrical overstress, and defective parts (e.g., resistors, capacitors, and cables). Integrated circuits may have defects of various kinds such as internal hard defects with secondary faults that manifest in the logic at the time of manufacturing, weaknesses that can cause failures later during operation by mechanisms such as electro-migration, or heat-induced mechanical stress that can cause fractures.

Installation and maintenance of computer-based systems consists primarily of placement and routing of components and cables, troubleshooting, and component replacement. Examples of how these activities can introduce faults include incorrect wiring or voltage, replacement of components with others with the wrong part number, obstructed airflow, debris that can cause short circuits or overheating, or physical damage to components from impact or mechanical stress during handling or installation.

The quality assurance process are the planned and systematic activities intended to provide adequate confidence that defects have been prevented or corrected in operational products. Quality assurance is applicable to every aspect of a system and can include activities to verify the processes for creating and sustaining a product, as well as activities to validate and verify the product itself. The level of rigor in quality assurance is determined by the criticality of a system or components relative to the desired attributes. The overall level of quality assurance effort is directly related to the level of rigor and the complexity of the product. The complexity is related to the number and variety of components and relations in the system. In general, for modern highly complex and integrated computational systems, quality assurance activities cannot guarantee perfect products, and thus, residual defects may remain in operational products. An analysis performed by Stecklein et al. shows that the cost to fix development faults grows exponentially thought the life cycle of a system [31]. Thus, from a cost perspective, the quality assurance process should aim to prevent and correct development mistakes, and non-operational faults in general, as close as possible to their time of occurrence.

2.2.2. Operational Threats

There are two kinds of operational threats: endogenous processes and exogenous processes. It is assumed here that the logic design of a safety-relevant system remains unchanged while the system is in operation. Hence, operational threats can only cause physical faults. Endogenous processes can cause physical faults by mechanisms such as electromagnetic interference, heat-induced stress, and random failures of integrated circuits. Exogenous process can cause faults mainly through violation of environmental assumptions that expose a system to stresses beyond its tolerance capabilities. Environmental condition standards such as RTCA DO-160 [32] define minimum test conditions representative of what an airborne system may encounter in operation. Some of the test conditions in DO-160 include temperature, humidity, vibration, sand and dust, fungus, magnetic effects, voltage spikes, radiated radio frequency (RF) electromagnetic energy, lightning, icing, and electrostatic discharge. High-energy particle radiation (e.g., neutrons) can also cause random faults and latch-up (i.e., internal short circuits caused by the activation of parasitic semiconductor structures) in integrated circuits [13].

2.2.3. Physical Faults

Physical faults (i.e., defects in the physical layer of a system) can be the primary and secondary effects of both non-operational and operational threats. Non-operational threat events in the form of design, implementation, and manufacturing errors are believed to be the most likely to cause multiple dependent physical faults, while operational threat events are usually assumed to cause random physical faults in single components. Quality assurance in non-operational life cycle phases is normally expected to provide adequate confidence that non-operational physical faults are non-existent or extremely unlikely. However, the physical fault arrival rate for individual electronics components due to operational threats tends to be constant and non-negligible in the domain of safety-relevant systems. The use of high quality components is generally not sufficient to mitigate the physical fault risks in safety-relevant systems, and redundancy-based mitigation strategies are required to achieve adequately low risk levels. Some primary physical faults may cause additional secondary physical faults as well as logical faults, which can then cause errors in the information layer and ultimately component and system failures. Special attention is required to mitigate the risk of non-operational and operational single threat events causing multiple faults as this situation can exceed the tolerance of redundancy-based mitigations and quickly exhaust spare resources [18]. Careful analysis and design is required to understand and minimize the physical fault introduction and propagation mechanisms, especially for unintended and unexpected coupling paths [33] [34].

2.2.4. Logical Faults

Logical faults are defects in the processes of computation and communication of data (i.e., defects in the logical layer of a system). These can be both primary and secondary effects of non-operational threats and secondary effects of operational threats. Logical faults are a special concern in complex and highly integrated systems as neither process-based nor product-based quality assurance, or a combination of the two, can guarantee complete absence of logical faults in deployed systems. A particular concern is that individual design and implementation threats can cause multiple faults in different components, including redundant components intended to mitigate fault risks. Common-cause faults require careful and deliberate consideration in safety-relevant systems whose overall quality assurance argument is supported by assumptions of fault independence and non-simultaneous activation of logical faults in redundant components. Here, too, careful analysis and design is required to understand and minimize the impact of logical faults caused by unexpected physical mechanisms [33] [34].

Additional information on logical faults, especially software faults, is available from many other sources. Chu et al. [35] present failure taxonomies for hardware and software in digital systems. Riecks et al. [36] describe a taxonomy with 16 failure classes and 114 fundamental failure types. Hayes [37] introduced a requirement fault taxonomy for software systems. Munson et al. [38] have proposed a quantifiable definition for software faults.

2.3. Logical Fault Activation and Effects Propagation

The service provided by an SOI consists of a sequence of service items (i.e., data items) that are characterized by a value and time of occurrence. An SOI experiences a service failure when the delivered service is incorrect. Under the failure model described here, an SOI service failure occurs when errors in the information layer propagate to the output ports of the SOI in the form of service items that are incorrect

in their value or time of delivery. These errors are caused by faults in the logical layer of the SOI.

The scope of logical fault effects depends on the structure and state of the SOI. A logical fault will remain dormant until the state of the system is favorable for the propagation of effects in the form of errors in the information layer. Errors can occur in control and data items and can impact the control flow and the data flow of the SOI. Errors can propagate through intended and unintended coupling paths, which are dependencies between elements in the SOI. Intended coupling paths involve the three basic functional behaviors in a system: data transformation, data storage, and data transfer. In addition, errors can result from unintended data or timing interactions between components, such as when there is sharing of system computation or communication resources.

Kopetz [39] defines safety as the probability that a system will survive for a given time interval without a critical failure mode (i.e., a failure mode that can lead to catastrophic consequences). Hence, from a safety perspective, we are interested in the probability distribution of SOI failure modes. In particular, we want the safety risk to be below a predetermined upper limit with an inverse relation between the likelihood and severity of failure modes. The development of a safety-relevant system is focused on ensuring that the failure characteristics of the SOI satisfy this risk constraint. The ability to prevent, detect, and contain the propagation of incorrect service items at the source and the users is a critical determining factor of failure modes and the safety risk level of the SOI.

2.4. Functional Failure Model

We are primarily concerned with safety-relevant systems whose failure can cause or contribute to damage or injury to people, property, and the natural environment. An SOI, which can be a top-level system or an internal component of a system, is designed to implement data processing functions allocated to it and realized as a service delivered to its environment in the form of a sequence of service items (i.e., data items), each characterized by a value and time of occurrence. Because an SOI only processes data, it cannot directly cause physical damage or injury. Thus, safety in the context of the SOI is about the operational and failure attributes of the delivered service from the perspective of potential effects in its operational environment.

This section and the next describe two levels of failure models intended for analysis of the propagation of failure effects. As functional and architectural design are primarily concerned with external and internal interface design, the failure models describe possible manifestations at interfaces. The **functional failure model** is from the perspective of a top-level system function at the external interface to the environment. The **component failure model** builds on the functional failure model by taking a structural system perspective with a source component delivering a service to multiple users (i.e., receivers). The models provide insight into the kinds of functional attributes that are critical in the analysis and mitigation of fault risks.

Figure 5 illustrates a high-level abstraction of the SOI as a service provider to its environment. The functional failure model is applicable to this level of abstraction of the SOI. We first consider an error model for individual service items. This is then applied in the definition of service failure modes as sequences of service items. This modeling approach is similar to the one presented by Powell [40].

Figure 5: Abstraction of SOI as Service Provider to its Environment

2.4.1. Source-Based versus Receiver-Based Failure Model

The SOI function can be modeled as being in one of three conditions:

- **Operational**: Operating as intended;
- **Loss of Function (LoF)**: Not operating; and
- **Malfunction (MF)**: Operating incorrectly (i.e., anomalous).

These basic conditions serve as a guide for other models presented in this report. Normally the functional condition is assessed at the source, but for the analysis of failure effects, it can be advantageous to assess the condition as perceived at the receiver. Recall from the event chain model, that the propagation of failure effects depends on the state of components downstream of the Initiating Unintended Event, which in this case is the functional failure of the SOI. There is a mapping, not necessarily one-to-one, between the functional condition at the source and the perceived condition at the receiver. In the following sections, the functional condition classification is based on the perception (i.e., the effect) at the receiver. This simplifies the architecture-level abstract modeling of first-order functional failure effects (i.e., at the immediate direct receiver). However, analyses of the causes of failure would have to consider the relation between source-based and receiver-based functional condition classification.

2.4.2. Functional Error Model

The classification of service item errors from a functional perspective is based on whether they can cause other errors in the environment. There are two types of errors:

- **Passive Error**: The value or time of the service item is erroneous, but this does not propagate additional (information) errors in the environment; and
- **Active Error**: The value or time of the service item is erroneous, and this may propagate additional (information) errors in the environment.

This classification is intended for abstract fault and failure effects analyses, and as such, it does not specify the nature of the errors or their effects propagation characteristics. Note that the definitions are based on the propagation of effects in the form of information errors in the environment. Time integration effects in dynamic environment processes are not in the scope of this definition. The model also does not specify if the containment of error effects is implicit or explicit, which depends on whether a deliberate action is taken at the SOI or the environment to prevent the propagation of undesired effects. Also, notice that the passive and active error classification is based on whether an error has the potential to propagate

effects, not whether effects will actually propagate. This convention is aligned with the conservative approach in safety-relevant systems.

Based on this error classification, a service item can be classified as:

- **Correct** (or **Valid**) **Item**: The value and time attributes of the service item are as intended;
- **Passive-Error Item**: The service item is erroneous in value or time, but it does not propagate errors in the environment; and
- **Active-Error Item**: The service item is erroneous in value or time and it may propagate errors in the environment.

It is implicit in the classification that a correct item does not propagate errors in the environment. Also, note that the classification is based on guarantees. A passive-error item is guaranteed not to propagate errors in the environment, but there is no guarantee for an active-error item.

2.4.3. Functional Failure Mode Model

Functional failure modes apply to sequences of service items. These failure modes can also be referred to as **functional health modes**. The categories of failure modes are:

- **Operational**: The service items are correct.
- **Failed-Passive**: The service items are correct, or passive-error, or a combination of these.
- **Failed-Active**: The service items are correct, passive-error, or active-error, or a combination of these.

This functional failure model defines a hierarchy of increasingly permissive behavior in which an operational function is the most constrained as it is defined (and guaranteed) as not having erroneous items, and failed-active mode is the least constrained and, in fact, arbitrary with possibly active-error service items. Figure 6 illustrates the implication graph for this failure mode model. From a safety perspective, the operational mode is the most predictable and least severe mode, and failed-active mode is unpredictable and the most severe mode due to a higher potential for severe effects in the environment. The failed-passive failure mode is a **controlled failure** and corresponds to a **Loss of Function** (LoF). The failed-active failure mode is an **uncontrolled failure** and corresponds to a **Malfunction** (MF).

Figure 6: Implication Graph for Functional Failure Modes

2.4.4. Transition Model for Functional Failure Modes

A functional failure is an event in which a function transitions out of the operational mode. In general, a safety-relevant system needs to withstand one or more faults in an operational mode before experiencing a failure. This operational fault tolerance is a fault-count safety margin against failure. When a functional failure event occurs, the severity of the failure depends on the characteristics of the environment and the mode and persistence (i.e., duration) of the failure. For a given failure mode, the **time to criticality** (**TTC**) (also called the **time to effect**, **TTE**) is the minimum duration to reach a particular level of severity of effects. If the system is able to **recover** (i.e., return to the operational mode) before the TTC, the effects of the failure event will not reach the worst-case level and the environment may recover back to normal or have a degraded but safe performance level. The TTC is a source of real-time functional performance constraints (e.g., hard deadlines) in safety-relevant systems. The TTC is effectively a time safety margin.

Figure 7 illustrates an example transition model for functional failure modes. In this figure, an operational safety margin of 0 means that a failure has occurred. The Down transitions in the Operational Mode represent internal faults and the Up transitions represent recovery events. Note that not all failures may be recoverable, and some failures may result in immediate severe consequences in the environment. From this figure, we can see that the primary characteristics for a failure scenario are: **time-to-failure**, **failure modes**, and **time-to-recover**. The time-to-failure is related to the operational fault tolerance by means of the **operational fault arrival rate** (i.e., the time rate at which operational faults are introduced) and the **fault activation rate** (i.e., the time rate at which existing faults begin to generate and propagate errors).

The illustration in the preceding Figure 5 represents a service relation between the SOI and its environment. This relation is a dependence of the environment on the service of the SOI such that performance and safety in the environment depends to some degree on the quality of the service delivered by the SOI. Avizienis et al. define **dependability** as the ability of a system to avoid service failures that are more frequent or more severe than is acceptable [41]. This means that a system is dependable if the risk of the dependence is lower than a maximum acceptable risk level (i.e., the boundary risk). The dependability of the SOI is determined by the three characteristics stated above: time-to-failure, failure modes, and time-to-recover. The time-to-failure characteristics determine the frequency of failures, and the failure mode and time-to-recover characteristics determine the severity of the failures.

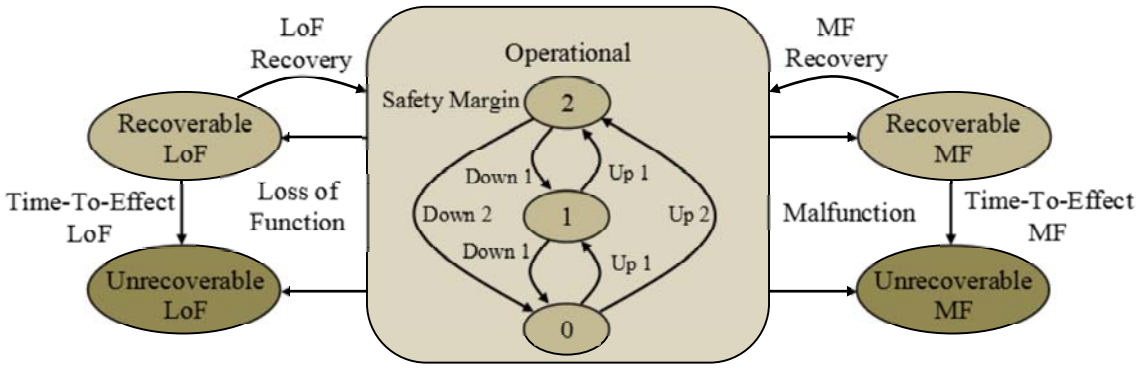


Figure 7: Example of Funtional Failure Mode Transition Model

2.5. Component Failure Model

The component failure model builds on the functional failure model by taking a structural system perspective. Here the source SOI is a system component that delivers a service to one or more user components. This is illustrated in Figure 8. The classification of SOI failure is based on the delivered service as perceived by the users through direct and independent inspection of service items at their respective communication interface.

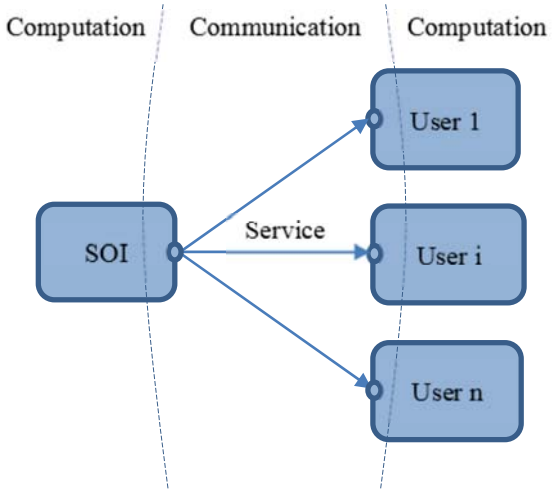


Figure 8: Structural View of the SOI Delivering a Service to Multiple Users

2.5.1. Component Error Model for Single-User Service

As illustrated in Figure 9, the component failure model assumes that user components monitor their inputs for errors using acceptance checks and that the data flowing down-stream includes the result of these checks. Based on this user model, we can define two major types of service item errors at an individual user:

- **Detectable Error**: The value or time of the service item is erroneous and this is detectable by the inline input error monitor at the user; and

- **Undetectable Error**: The value or time of the service item is erroneous, but this is not detectable by the inline input error monitor at the user.

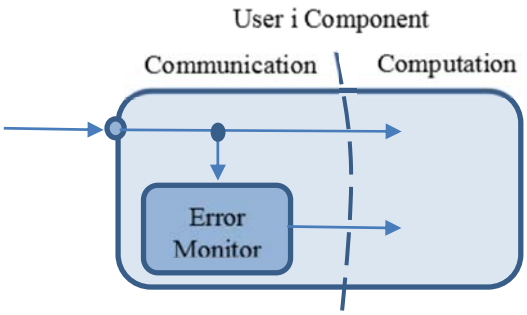


Figure 9: Assumed Error Monitor at Input Port of User Component

The detectability of an error determines the response of the user. Implicit in the definition of detectable error is the exclusion of false-positive detection for correct items, which means that a correct item is not rejected by the input acceptance checks. It is assumed that a missing item is detectable. It is also assumed that a user will react to a detectable item error, whether it is a missing item or otherwise, by rejecting the item and taking action to contain the propagation of undesired effects. No such assumption is made about the containment of undetectable-error effects. Because of this, a detectable error is also called an **omissive error** and an undetectable error is called a **transmissive error**.

Using these error types, a service item can be classified as follows:

- **Correct Item (C)**: The value and time attributes of the item are as intended;
- **Detectable-Error Item (D)**: The service item is erroneous in value or time and the errors are detectable; and
- **Undetectable-Error Item (U)**: The service item is erroneous in value or time, but the errors are undetectable.

Detectable-error items and undetectable-error items are also called simply **detectable** items and **undetectable** items, respectively. A detectable item is also called an **omissive item** and an undetectable item is called a **transmissive item**. Correct, detectable, and undetectable items are denoted C, D, and U, respectively. Correct C items and undetectable U items are called **acceptable** items, as they are not rejected (i.e., they are accepted) by the inline error monitor at the user.

2.5.2. Component Failure Mode Model for Single-User Service

Failure modes apply to sequences of service items. The categories of service modes for single-user service are:

- **Correct (C)**: The service items are correct.
- **Detectable (D)**: The service items are be correct or detectable, or a combination of these.

- **Undetectable (U)**: The service items are correct, detectable, or undetectable, or a combination of these.

Notice that this classification defines increasing levels of permissiveness (and hence, uncertainty) in the items, such that the Detectable failure mode has more possible options (i.e., is less constrained) than Correct and the Undetectable failure mode is effectively arbitrary (i.e., it can have C, D, and U items). The potential complexity of the failure modes increases with the increase in permissiveness. Also, notice that the items are called and denoted based on their worst-case type of error relative to the presence of erroneous items and possible effects at the user component. Figure 10 illustrates the implication relation for the failure modes such that the Correct mode category is contained in the Detectable mode category, which is itself contained in the Undetectable mode category.

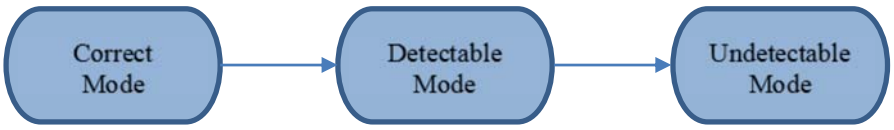


Figure 10: Implication Graph for Single-User Service Failure Modes

2.5.3. Component Error Model for Multiple-User Service

Multiple-user service is illustrated in Figure 8, where the number of user components is denoted *n*. A multiple-user service item consists of *n* replicated single-user items. The error model presented here for multi-user service is the **Omissive-Transmissive Hybrid (OTH)** fault model [42] [43]. Multiple-user service introduces the dimension of symmetry (i.e., consistency) to the characterization of service items. A service item is **symmetric** if the items perceived by all the users are equal. **Equality** can be exact or approximate (i.e., bounded difference) depending on the element of the service item (i.e., value or time) and the semantics (i.e., meaning) of the service. Asymmetry among the users is a form of error and has the potential to propagate error effects in a system. Generally, an asymmetric multi-user service item is more severe than a symmetric item.

A multiple-user service item can be expressed as a vector $X = (x_1; x_2; \ldots; x_n)$, where x_i is the single-user item at user i. In the OTH model, it is assumed that correct single-user items C at different users are equal (i.e., correct items are symmetric). Likewise, all detectable single-user items D are equal. Pairs of undetectable single-user items U may be either equal or unequal. Unequal U items are denoted by different sub-scripts (e.g., $U_i \neq U_j$). A correct C item and an undetectable U item may be either equal or unequal in value and/or time, depending on the semantics, but they are not the same with respect to correctness[1]. The OTH error categories for a multiple-user service item are the following:

- **Correct (C)**: Every user accepts a correct single-user service item.

- **Omissive Symmetric (OS)**: Every user rejects its single-user service item.

- **Transmissive Symmetric (TS)**: Every user accepts an undetectable single-user service item, but all

[1] For example, if integer values smaller than 5 are correct, values from 6 to 9 are incorrect but undetectable, and values that differ by 3 or less are equal, then a value of 4 is correct and 6 is undetectable, but 4 and 6 are equal.

the single-user service items are equal.

- **Strictly Omissive Asymmetric (SOA)**: Some users reject their single-user service item and others accept a correct item.

- **Single-Data Omissive Asymmetric (SDOA)**: Some users reject their single-user service item and others accept an undetectable item, but all the accepted items are equal.

- **Transmissive Asymmetric (TA)**: The users have other patterns of service items.

To gain insight into the OTH classification, we consider two levels of symmetry in multiple-user service items. The first type of symmetry covers all the single-user items taken together. We have **overall symmetry** if all the users receive equal service items, which can be C, D, or U. Otherwise, the item has **overall asymmetry**. These conditions are denoted S_n and A_n, respectively, as they apply to all n users taken together. The second type of symmetry is item equality across accepting users (i.e., users with C or U items), which can range in number from 0 to n. A multiple-user item has **acceptable-item symmetry** if the accepting users have equal items. Otherwise, the multiple-user item has **acceptable-item asymmetry**. These are denoted S_A and A_A, respectively. Notice that S_n implies S_A, and A_A implies A_n.

The OTH classification can be decomposed based on three criteria: worst-case single-user error, overall symmetry, and acceptable-item symmetry. The worst-case single-user error specifies the worst-case single-user item at anyone of the users based on the severity ranking C < D < U (i.e., C is less severe than D and D is less severe than U). C_L denotes the condition that the worst-case single-user item error is C, D_L denotes a worst-case of D, and U_L denotes a worst-case of U. For the overall and acceptable-item symmetry conditions, asymmetry is a more severe error than symmetry. That is: $S_n < A_n$ and $S_A < A_A$. There is no clear way to compare the relative severity of single-user item errors and symmetry errors. However, acceptable-item symmetry is clearly a stronger determinant of severity than overall symmetry, as the only way to have acceptable-item asymmetry is by having at least one undetected single-user error, which by definition has the potential to propagate errors.

Table 2 shows the decomposition of the OTH classification. Figure 11 shows the severity relation for the OTH classification based on the severity of errors for the various decomposition criteria. This is a partial-order relation as SOA and TS differ in the worst-case single-user error and the overall symmetry condition and there is no basis to rank the relative severity of these dimensions. Note that TA error is the only kind with acceptable-item asymmetry, which is the most severe kind.

Table 2: Decomposition of Omissive-Transmissive Hybrid Error Model by Error Dmensions

OTH Category	Worst-case Single-User Error	Overall Symmetry Condition	Acceptable-Item Symmetry Condition	Example Patterns For n = 4
Correct (C)	C_L	S_n	S_A	(C; C; C; C)
Omissive Symmetric (OS)	D_L	S_n	S_A	(D; D; D; D;)
Strictly Omissive Asymmetric (SOA)	D_L	A_n	S_A	(C; C; D; D)
Transmissive Symmetric (TS)	U_L	S_n	S_A	(U; U; U; U)
Single-Data Omissive Asymmetric (SDOA)	U_L	A_n	S_A	(U; U; D; D)
Transmissive Asymmetric (TA)	U_L	A_n	A_A	(C; U; D; D) (U_1; U_2; D; D) (U_1; U_1; U_2; U_3)

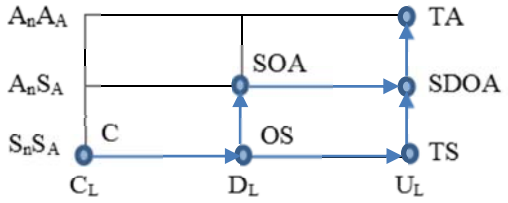


Figure 11: Severity Relation for OTH Error Classification

2.5.4. Component Failure Modes for Multiple-User Service

The failure modes for multiple-user service apply to sequences of service items and characterize the health status of the source component. In this model, the failure modes are based on the OTH error model and the severity relation in Figure 11. These **OTH failure modes** are defined as sets of permissible OTH errors and the names correspond to the most severe error in their set. The categories of failure modes are:

- **Correct (C)**: The service items are correct.
- **Omissive Symmetric (OS)**: The service items are correct or omissive symmetric, or a combination of these.
- **Transmissive Symmetric (TS)**: The service items are correct, omissive symmetric, or transmissive symmetric, or a combination of these.
- **Strictly Omissive Asymmetric (SOA)**: The service items are correct, omissive symmetric, or strictly omissive asymmetric, or a combination of these.
- **Single-Data Omissive Asymmetric (SDOA)**: The service items are correct, omissive symmetric, transmissive symmetric, strictly omissive asymmetric, or single-data omissive asymmetric, or a combination of these.
- **Transmissive Asymmetric (TA)**: The service items are arbitrary (i.e., any combination of any items in the OTH classification). This failure mode corresponds to a Byzantine fault [44] [33].

The severity relation for OTH failure modes is the same as for OTH errors (see Figure 11). Also, notice that as the severity of the modes increases so does the permissiveness and uncertainty about the error category of the service items (i.e., higher severity modes are less constrained). Figure 11 also captures the implication relation for the OTH failure modes.

2.5.5. Failure Models with Explicit Value and Time Dimensions

It is possible to refine the component error models and failure mode models by introducing specific errors for the value and time elements of service items. A good initial approach for this would be to leverage the value and time error models proposed by Powell [40]. Avizienis et al. [13] provide additional

information on service failure modes. The refinement of the service error models is beyond the scope of this report. Formal mathematical definitions of failure models are given by, for example, Powell [40] and Warns [45].

3. Final Remarks

This document presented an overview of threats to safety-relevant computer-based systems. Models of errors and failure modes for functions and components were also presented. This document is intended to be part of a design guide for safety-relevant computer-based systems. The next report in this series will address the topic of threat-risk mitigation.

References

[1] S. C. Beland, "Assuring a Complex Safety-Critical Systems of Systems," *AeroTech Congress & Exhibition, SAE Technical Paper 2007-01-3872,* September 2007.

[2] D. Jackson, M. Thomas and L. I. Millett, Eds., *Software for Dependable Systems: Sufficient Evidence?,* Committee on Certifiably Dependable Software Systems; National Research Council, 2007.

[3] National Research Council, Steering Committee for the Decadal Survey of Civil Aeronautics, *Decadal Survey of Civil Aeronautics: Foundation for the Future,* 2006.

[4] *National Aeronautics Research and Development Plan,* Office of Science & Technology Policy, 2010.

[5] D. C. Winter, *Statement before a hearing on Networking and Information Technology Research and Development (NITRD) Program,* Committee on Science and Technology, U.S. House of Representatives, 2008.

[6] A. Mozdzanowska and R. J. Hansman, *System Transition: Dynamics of Change in the US Air Transportation System,* International Center for Air Transportation, Massachusetts Institute of Technology, Report Number ICAT-2008-3, 2008.

[7] Joint Planning and Development Office (JPDO), *NextGen Integrated Development Plan, Version 1.0,* 2008.

[8] Aerospace Vehicle Systems Institute (AVSI), *AFE #58 Summary Final Report,* System Architecture Virtual Integration (SAVI) Program, Document SAVI-58-00-01, version 1, 2009.

[9] A. Avizienis, "The Four-Universe Information System Model for the Study of Fault Tolerance," in *Proceedings of the 12th Annual International Symposium on Fault-Tolerant Computing,* Santa Monica, California, 1982.

[10] Federal Aviation Administration (FAA), "FAA System Safety Handbook," 2000.

[11] B. Hall and K. Driscoll, "Distributed System Design Checklist," NASA Langley Research Center, NASA/CR-2014-218504, 2014.

[12] C. Scherrer and A. Steininger, "Dealing With Dormant Faults in an Embedded Fault-Tolerant Computer System," *IEEE Transactions on Reliability,* vol. 52, no. 4, pp. 512 - 522, December 2003.

[13] A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing,* Vols. 1, No. 1, pp. 11 - 33, January - March 2004.

[14] B. Parhami, "From Defects to Failures: a View of Dependable Computing," *ACM SIGARCH Computer Architecture News – Special Issue on Architectural Support for Operating System,* vol. 16, no. 4, pp. 157 - 168, September 1988.

[15] B. Parhami, "A Multi-Level View of Dependable Computing," *Computers and Electrical Engineering,* vol. 20, no. 4, pp. 347 - 368, July 1994.

[16] SAE International, *Aerospace Recommended Practice: Fault/Failure Analysis for Digital Systems and Equipment (ARP1834A),* SAE International, 1992.

[17] S. C. Beland and B. BonJour, "Functional Failure Path Analysis of Airborne Electronic Hardware," in *The 19th Digital Avionics Systems Conference,* Philadelphia, PA, 2000.

[18] SAE International, *Aerospace Recommended Practice: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment (ARP4761),* 1996.

[19] N. G. Leveson, Safeware: System Safety and Computers, Addison-Wesley, 1995.

[20] IAEA, "Protecting against Common Cause Failues in Digital I&C Systems of Nuclear Power Plants," International Atomic Energy Agency (IAEA), 2009.

[21] A. Mosleh, K. N. Fleming, G. W. Parry, H. M. Paula, D. H. Worledge and D. M. Rasmuson, "Procedures for Treating Common Cause Failures in Safety and Reliability Studies: Analytical Background and Techniques," U.S. Nuclear Regulatory Commission, 1989.

[22] T. E. Wierman, D. M. Rasmuson and A. Mosleh, "Common-Cause Failure Database and Analysis System: Event Data Collection, Classification, and Coding," U.S. Nuclear Regulatory Commission, 2007.

[23] A. Mosleh, D. M. Rasmuson and F. M. Marshall, "Guidelines on Modeling Common-Cause Failures in Probabilistic Risk Assessment," U.S. Nuclear Regulatory Commission, 1998.

[24] P. J. Rutledge, "An Analysis of Dependent Failures in Spacecraft: Root Causes, Coupling Factors, Defenses, and Design Implications," University of Maryland at College Park, 1997.

[25] P. J. Rutledge and A. Mosleh, "Dependent-failures in spacecraft: root causes, coupling factors, defenses, and design implications," in *Proceedings of the 1995 Annual Reliability and Maintainability Symposium*, Washington, D.C., 1995.

[26] J. E. Stott, P. Britton, R. W. Ring, F. Hark and G. S. Hatfield, "Common Cause Failure Modeling: Aerospace versus Nuclear," in *10th International Probabilistic Safety Assessment and Management Conference*, Seattle, WA, 2010.

[27] L. M. Kaufman, S. Bhide and B. W. Johnson, "Modeling of Common-Mode Failures in Digital Embedded Systems," in *Proceedings of the 2000 Annual Reliability and Maintainability Symposium*, Los Angeles, CA, 2000.

[28] N. Suri, C. J. Walter and M. M. Hugue, "Introduction," in *Advances in Ultra-Dependable Distributed Systems*, IEEE Computer Society Press, 1995, pp. 3 - 15.

[29] W. L. Heimerdinger and C. B. Weinstock, "A Conceptual Framework for System Fault Tolerance," Carnegie Mellon University - Software Engineering Institute (CMU/SEI), 1992.

[30] M. M. Hugue, "Fault Type Enumeration and Classification," Office of Naval Research, 1991.

[31] J. M. Stecklein, J. Dabney, B. Dick, B. Haskins, R. Lovell and G. Moroney, "Error Cost Escalation Through the Project Life Cycle," in *14th Annual International Symposium International Council on Systems Engineering (INCOSE)*, San Diego, CA, 2004.

[32] RTCA, Inc., *Environmental Conditions and Test Procedures for Airborne Equipment (RTCA DO-160G)*, 2010.

[33] K. Driscoll, B. Hall, H. Sivencrona and P. Zumsteg, "Byzantine Fault Tolerance, from Theory to Reality," in *The 22nd International Conference on Computer Safety, Reliability and Security SAFECOMP*, 2003.

[34] K. Driscoll, "Real System Failures: Observed Failure Scenarios," September 2013. [Online]. Available: https://c3.nasa.gov/dashlink/resources/624/. [Accessed 18 June 2015].

[35] T.-L. Chu, M. Yue and W. Postma, "A Summary of Taxonomies of Digital System Failure Modes Provided by thge DigRel Task Group," Brookhaven National Laboratory, 2012.

[36] J. Riecks, W. Storm and M. Hollingsworth, "Concept Development for Software Health Management," NASA, 2011.

[37] J. H. Hayes, "Building a Requirement Fault Taxonomy: Experiences from a NASA Verification and Validation Research Project," in *ISSRE '03 Proceedings of the 14th International Symposium on Software Reliability Engineering*, 2003.

[38] J. C. Munson, A. P. Nikora and J. S. Sherif, "Software faults: A quantifiable Definition," *Advances in Engineering Software*, vol. 37, no. 5, pp. 327 - 333, May 2006.

[39] H. Kopetz, Real-Time Systems: Design Principles for Distributed Embedded Applications, 2nd ed., Springer Science + Business Media, 2011.

[40] D. Powell, "Failure Mode Assumptions and Assumption Coverage," in *Twenty-Second International Symposium on Fault-Tolerant Computing (FTCS-22)*, Boston, MA, 1992.

[41] A. Avizienis, J. C. Laprie, B. Randell and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11 - 33, January - March 2004.

[42] M. H. Azadmanesh and R. M. Kieckhafer, "Exploting Omissive Faults in Synchronous Approximate Agreement," *IEEE Transactions on Computers*, vol. 49, no. 10, pp. 1031 - 1042, October 2000.

[43] M. H. Azadmanesh and R. M. Kieckhafer, "New Hybrid Fault Models for Asynchronous Approximate Agreement," *IEEE Transaction on Computers,* vol. 45, no. 4, pp. 439 - 449, April 1996.

[44] L. Lamport, R. Shostak and M. Pease, "The Byzantine Generals Problem," *ACM Transactions on Programming Languages and Systems (TOPLAS),* vol. 4, no. 3, pp. 382-401, 1982.

[45] T. Warns, Structural Failure Models for Fault-Tolerant Distributed Computing, Germany: Springer, 2010.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 09 - 2015	Technical Memorandum	

4. TITLE AND SUBTITLE

Overview of Threats and Failure Models for Safety-Relevant Computer-Based Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Torres-Pomales, Wilfredo

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

999182.02.50.07.02

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

L-20586

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA-TM-2015-218783

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 62
Availability: NASA STI Program (757) 864-9658

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This document presents a high-level overview of the threats to safety-relevant computer-based systems, including (1) a description of the introduction and activation of physical and logical faults, (2) the propagation of their effects, and (3) function-level and component-level error and failure mode models. These models can be used in the definition of fault hypotheses (i.e., assumptions) for threat-risk mitigation strategies. This document is a contribution to a guide currently under development that is intended to provide a general technical foundation for designers and evaluators of safety-relevant systems.

15. SUBJECT TERMS

Computer; Error; Failure; Fault; Model; Safety; System

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	30	(757) 864-9658